



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/449,021	11/24/1999	HELMUT EMMELMANN	EMMEL-1000US0	5718
28554 7590 10/13/2010 Vierra Magen Marcus & DeNiro LLP 575 Market Street, Suite 2500 San Francisco, CA 94105				
EXAMINER				
KENDALL, CHUCK O				
ART UNIT		PAPER NUMBER		
2192				
MAIL DATE		DELIVERY MODE		
10/13/2010		PAPER		

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 09/449,021
Filing Date: November 24, 1999
Appellant(s): EMMELMANN, HELMUT

Richard A. Nebb Reg. No. 33,540
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 06/09/10 appealing from the
Office action mailed 01/21/10.

(1) Real Party in Interest

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

The copy of the appealed claims contained in the Appendix to the brief is correct.

For the above reasons, it is believed that the rejections should be sustained.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

WEBWRITER I	Arturo et al.	1996
WEBWRITER II	Arturo et al.	1997

(9) Grounds of Rejection

An Interview between Examiner and Attorney of record (Richard Nebb) was conducted on 09/24/10 and based on the discussed subject matter pointed out in the Interview summary the following claims having such subject matter are now allowable or at least objected to based on the outstanding rejections on the based claims.

Claims 6, 8, 51 – 58, 74 – 96 and 114 – 128 are now allowable.

Claims 3 – 5, 24, 25, 27 – 29, 31, 43, 64 – 66, and 70 are objected to based on the outstanding rejections on the base claims.

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1, 22, 23, 26, 30, 32, 41, 42, 59 – 63, 67 – 69, 71 – 73 are rejected under 35 U.S.C. 103(a) as being unpatentable over Web Writer: A Browsers -Based Editor for Constructing Web Applications Arturo et al. Published May 1996 (Hereinafter “Web Writer”) in view of Web Writer II, Arturo et al. Published 1997.

Any Arguments regarding the current status of claims as outlined above in reference to any previous arguments made by Applicant in any one of Examiner’s previous rejections are no longer relevant as such claims have either pointed out as having allowable subject matter or remain objected to based on rejected base claims.

A Full copy of the previous rejection is being reproduced below for completeness:

Claim Rejections - 35 USC § 103

3. *The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:*

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

4. *Claims 1 – 6, 8, 22 – 33, 41 – 43, 51 – 96, and 114 – 128 are rejected under 35 U.S.C. 103(a) as being unpatentable over Web Writer: A Browsers -Based Editor for Constructing Web Applications Arturo et al. Published May 1996 (Hereinafter "Web Writer") in view of Web Writer II, Arturo et al. Published 1997.*

Regarding claims 1, 6, 22, 26, 51, 59, 74, 90, 114, 125, a computer-readable medium encoded with computer programs having executable instructions software development system for editing software applications that run on a data network and a client computer, wherein the client computer runs a browser program, and whereupon request by the browser program at least one of the applications generates generated documents for display by the browser program on a display device and responds to the request with the generated documents page

comprising:

a page document generator program running at least part of one of the applications being edited and generating the generated documents said generated documents including additional editing features for interpretation by the browser program (page 9, column 1, see Web Writer page Generator); and

an editor program dynamically operating on the generated documents pages displayed by the browser program via the editing features (page 9, column 1, see Web Writer page Generator, see on the fly, also see page 2, first paragraph for buttons and computing content on the fly for components as claimed in claim 59).

Although, Web Writer doesn't disclose having a server, Web WriterII discloses Web Writer coupling a WebServer in its implementation (section 1.2). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made combine Web Writer and Web Writer II, because it would enable making changes on the fly remotely.

Regarding claim 2, a computer-readable medium software development system as claim 1, further encoded with further comprising a plurality of components, and wherein developed the software applications comprise at least one page document template capable of containing components, and wherein the editor provides features to insert, modify and delete a component on at least one page document template, and wherein the page-document generator executes selected components on page document templates (page 2, Column 1, see Heading

The Web Writer Application Model, shows template page).

Regarding claim 3, a computer-readable medium software development system as in claim 2, wherein at least one of the components reacts interactively on user input by executing instructions of said component on the server computer (see, FIG. 2).

Regarding claim 4, a computer-readable medium software development system as in claim 3, wherein at least one of the components contains at least one other component (FIG. 2, also see FIG. 4).

Regarding claim 5, a computer-readable medium software development system as in claim 3, wherein the set of components on pages documents generated from at least one page document template can vary for different requests of said page document template (page 2, Column 1, see Heading The Web Writer Application Model, shows template page, also see page 9).

Regarding claim 8, the development system of claim 6, wherein a component is nested within a component (Web Writer, see FIG. 4).

Regarding claim 23, a computer readable medium as in claim 22, wherein the editor program operates functional applications in an edit mode permitting editing directly in the web browser (page 2, Column 1, see Heading The Web Writer Application Model, shows template page).

Regarding claim 24, a computer readable medium as in claim 23 wherein at least one of the components contains instructions and can react on subsequent document requests containing user responses by executing selected instructions of said component (See Web Writer page 2 – 9 and et. seq).

Regarding claim 25, a computer readable medium as in claim 24, of component classes, each component class implementing one component kind (Web Writer page 2 – 9, et seq); and

a parser program able to detect components marked on document templates (Web Writer page 2 – 9, et seq);

wherein the document generator program works upon a document request using component classes to generate browser code (Web Writer page 2 – 9, et seq); and

wherein the editor program is capable of showing a menu of components for insertion into the document templates (See Web Writer page 2 – 9 see editor, equivalent function).

Regarding claim 27, the system of claim 26, wherein the first document includes at least one component being executed by the first software program (See Web Writer page 2 – 9 and et. seq).

Regarding claim 28, the system of claim 27, wherein the second document includes handles and choosing one of the handles selects a component for an

editing operation (Web Writer page 4, see clicking on its handles).

Regarding claim 29, the system of claim 28, wherein at least one handle indicates the position of at least one component contained in the first document and said editing operation includes modifying the component, deleting the component, and displaying information regarding the component (Web Writer page 4, see clicking on its handles).

Regarding claim 30, the system of claim 26, wherein the features include scripts (See Web Writer page 2 – 9 and et. seq).

Regarding claim 31, the system of claim 30, wherein the scripts encapsulate information from the first document (See Web Writer page 2 – 9 and et. seq)

Regarding claim 32, the system as in claim 26, wherein the features incorporate information regarding the first document into the second document (See Web Writer page 2 – 9 and et. seq).

33. (currently amended) A system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server computer (Web WriterII, see page 1510 section 4.1).

Regarding claim 41, (currently amended) a computer-readable medium as in claim 1, the editor program comprising a client part for execution on the client

computer (Web WriterII, see page 1510 section 4.1).

Regarding claim 42, a computer-readable medium as in claim 41, wherein the client part comprises instructions for execution during editing that are automatically downloaded from the server computer in a request prior to editing (Web WriterII, see page 1510 section 4.1).

Regarding claim 43, a system computer-readable medium as in claim 26, additionally comprising at least one script for automatic download to the client that works in cooperation with the second document to permit editing of the first document (Web WriterII, see page 1510 section 4).

52. (previously amended) The system of claim 51, wherein at least some components include fourth program instructions including steps to generate browser code for transmission to the first software program in response to a request from the first software program, wherein the browser code can differ for multiple requests for the same document template (See Web Writer page 2 – 9 and et. seq).

53. (currently amended) A system as in claim 52 having a data network, coupling the computer and a client computer, the first program running on the client computer (Web WriterII, see page 1508 section 1.2).

54. (currently amended) A system as in claim 52 wherein second documents include HTML pages with embedded scripts(See Web Writer page 2 – 9 and et. seq).

55. (currently amended) The system of claim 52, wherein the editing functions includes adding a component to a document template, removing a component from a document template, and modifying a component on a document template (See Web Writer page 2 – 9 , see editor).

56. (previously added) The system of claim 52, further comprising a fifth software program used by the second software program while processing selected document requests, the fifth software program including fifth instructions for generating generated documents from document templates thereby calling fourth program instructions (See Web Writer page 2 – 9 , see templates and generator).

57. (previously added) The system of claim 56, wherein the generated document includes, if requested in edit mode, edit features for interpretation by the first software program (See Web Writer page 2 – 9 , see editor).

58. (previously amended) The system of claim 56 further comprising instructions to allow the user to click on the generated document to select items to perform edit functions on (See Web Writer page 2 – 9 , see editor).

60. (currently amended) The software development system of Claim 59 comprising

a data network which couples a server computer and a client computer, the document generator program running on the server computer, the editor program at least partly running on the client computer(Web WriterII, see page 1510 section 4).

61. (currently amended) The software development system of claim 60 comprising fourth instructions for execution during document generation to collect edit-information for use by the editor program (See Web Writer page 2 – 9 , see editor).

62. (currently amended) The software development system of claim 60, wherein the editor program uses a web browser for displaying said view(See Web Writer page 2 – 9, shows web browser).

63. (currently amended) The software development system of claim 60, comprising instructions for automatically repeating the requesting that the document generator processes the dynamic web document when if required (See Web Writer page 2 – 9, shows web browser).

64. (currently amended) The software development system of Claim 59 further comprising a plurality of components including at least one component marked on said dynamic web document and including instructions for use by the document generator program to generate browser code (See Web Writer page 2 – 9, see on the fly).

65. (currently amended) The software development system of claim 64, wherein the editor program uses a web browser for displaying said view (See Web Writer page 2 – 9, shows web browser).

66. (currently amended) The software development system of claim 64, wherein the modification functions includes insertion of a component, ~~detete~~ deletion of a component, and ~~modify~~-modification of a component (See Web Writer page 2 – 9, shows see editor).

67. (previously added) The software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document(See Web Writer page 2 – 9, shows see editor).

68. (currently amended) The software development system of claim 59 comprising sixth instructions to collect edit-information for use by the editor program, said sixth instructions for execution during document generation(See Web Writer page 2 – 9, shows document generator).

69. (currently amended) The software development system of claim 68, wherein the editor program uses the edit-information to correctly modify the dynamic web document (See Web Writer page 2 – 9, shows see editor).

70. (currently amended) The software development system of claim 69, further comprising a plurality of components wherein the edit-information comprises position information on selected components marked on the dynamic web document (See Web Writer page 2 – 9, shows dynamic web content).

71. (currently amended) The software development system of claim 59, wherein the editor program uses a web browser for displaying said view(See Web Writer page 2 – 9, shows see editor).

72. (currently amended) The software development system of claim 71, wherein the first instructions comprise seventh instructions for initiating a reload in the browser (See Web Writer page 2 – 9, shows browser).

73. (currently amended) The software development system of claim 59 wherein the editor program comprises eighth instructions to display information on at least one element of at least one dynamic web document, that is replaced during document generation, without requesting that the document generator program generates a document (See Web Writer page 2 – 9, shows generator).

75. (currently amended) The system as in claim 74, wherein the editing functions comprises adding a component, modifying a component, and deleting a component (See Web Writer page 2 – 9, shows see editor).

76. (currently amended) The ~~software development~~ system as in claim 74, wherein tag syntax is used to denote at least one component on at least one document template, whereby the tag name identifies the component kind(See Web Writer page 2 – 9, shows use of templates).

77. (currently amended) The system of claim 74 comprising a server computer

coupled to a client computer by a data network, the document generator program running on the server computer, and the editor program running, at least partly, on the client computer (Web WriterII shows Web server and editor, also see Web Writer).

78. (currently amended) The ~~software-development~~-system as in claim 74, wherein at least one component; that can react interactively on subsequent document requests can be excluded from said generated document upon selected document requests for said document template (See Web Writer page 2 – 9, shows generator and templates).

79. (currently amended) The ~~software-development~~-system as in claim 78 comprising third instructions to prevent excluded components from reacting on subsequent document requests (See Web Writer page 2 – 9).

80. (currently amended) A ~~software-development~~-system as in claim 79, said third instructions comprising fourth instructions to, upon a first document request, store information in session memory on some of the components that are present on the document generated based on the first document request, and fifth instructions to, upon subsequent document requests, only react on components that have been remembered in session memory thereby avoiding tampering with excluded components on the side of the first program (See Web Writer page 2 – 9).

81. (currently amended) A ~~software-development~~ system as in claim 74 wherein at least one first component contains sixth instructions to decide upon a request for said document template about exclusion of components nested inside the first component from the generated document (See Web Writer page 2 – 9, shows Templates).

82. (currently amended) A ~~software-development~~ system as in claim 74 the system ~~able to provide~~ providing an editable view taking the varying set of components into account (See Web Writer page 2 – 9, shows editor and display/interface).

83. (currently amended) A ~~software-development~~ system as in claim 74 ~~system able to provide~~ providing an editable view that includes and excludes selected components on different requests for said document template ~~similarly as~~ similar to the end user's view of the document template (See Web Writer page 2 – 9, shows editor and template).

84. (currently amended) A ~~software-development~~ system as in claim 74 wherein a document generated for at least one document template contains more components than the document template for at least one document request (See Web Writer page 2 – 9, shows templates).

85. (currently amended) The ~~software-development~~ system as in claim 74, wherein

multiple instances of at least one third component denoted on the document template can be included in at least one of the documents generated from said document template(See Web Writer page 2 – 9, shows templates).

86. (currently amended) The system as in claim 74, comprising seventh instructions to assign a unique identifier to each component instance of at least one seventh component, whereby the seventh component includes eighth instructions to qualify names generated into the browser code with the unique identifier (See under the heading, The Web Writer Application Model, " a set of named form elements used as variables").

87. (currently amended) A system as in claim 74, wherein at least one fourth component contains ninth instructions to decide how many instances of components nested inside the fourth component are included in the documents generated from said document template (See under the heading, The Web Writer Application Model, one or more dynamic areas on each template page," Examiner interprets this to be nested components).

88. (currently amended) A system as in claim 74 the editor program able to provide an editable view that includes multiple instances of selected components similar to the end user's view of the document template (See under the heading, Scripts and Formatting for Output areas, multiple instances of variables are disclosed).

89. (currently amended) A system as in claim 74 wherein at least one sixth component includes tenth instructions to display the sixth component, the tenth instructions being used to generate browser code for displaying the sixth component during editing as well as during normal use of the component (See Web Writer page 2 – 9, shows browser).

91. (currently amended) The system as in claim 90 wherein the editor program includes instructions to display at least two windows, a first browser window displaying said document and a second window for displaying information on an element contained in said document (See Web Writer page 2 – 9).

92. (currently amended) The system as in claim 90 comprising a second software program having instructions for storing modifications on said document in cooperation with the first software program (See Web Writer page 2 – 9, see editor).

93. (currently amended) The system ~ as in claim 92 further comprising a third program having instructions for transforming an original document into the document, the browser displaying the document as said view looking similar to the original document and interpreting editing features contained in the document (See Web Writer page 2 – 9, see editor).

94. (currently amended) The system ~ as in claim 93 wherein said original document is a dynamic document having components denoted thereon, the third

software program comprising instructions for generating browser code in cooperation with selected instructions contained in the components (See Web Writer page 2 – 9, shows browser).

95. (currently amended) The system ~ as in claim 94 comprising a client computer connected to a server computer via a data network, wherein the browser together with the first software program is running on at the client computer and the second and the third software program run on the server computer (See Web Writer page 2 – 9).

96. (currently amended) The system as in claim 90 wherein links contained in said document stay functional allowing the user to browse and edit at the same time (See Web Writer page 2 – 9, see editor).

115. (previously amended) The system of claim 128 wherein first features include fourth program instructions for passing information to the editor (See Web Writer page 2 – 9, see editor).

116. (currently amended) The system of claim 115 wherein at least part of said information is collected during execution of selected components on the server compute r(See Web Writer page 2 – 9).

117. (currently amended) The system of claim 115 wherein said information is transmitted from the server computer to the client computer (See Web Writer II et seq.).

118. (previously amended) The system of claim 115 wherein said information includes attribute values of said component (See Web Writer page 2 – 9, see editor).

119. (previously amended) The system of claim 128 wherein first features include fifth instructions that display additional editing features of the component during editing (See Web Writer page 2 – 9, see editor).

120. (previously added) The system of claim 119 wherein said editing features include handles (See Web Writer page 2 – 9, see editor).

121. (previously amended) The system of claim 128 wherein first features include an extension for use by the editor, said extension for enabling editing of an attribute value of the components (See Web Writer page 2 – 9, see editor).

122. (currently amended) The system of claim 121 wherein said extension enables display of a page for editing the-attribute values of the components (See Web Writer page 2 – 9, see editor).

123. (previously amended) The system of claim 128 wherein at least one component is denoted on at least one document template using tag syntax, whereby the tag name identifies a component kind (See Web Writer page 2 – 9, see templates).

124. (previously amended) The system of claim 114 containing at least one component wherein second program instructions are used to generate browser code for displaying the component during editing and during normal use (See Web Writer page 2 – 9, see browser).

126. (currently amended) The method of claim 125 wherein the running step and the displaying step are repeated after the modification function (See Web Writer

page 2 – 9, see editor).

127. (previously added) The method of claim 125 further comprising collecting edit information for use by the identifying step (See Web Writer page 2 – 9, see editor).

128. (previously added) The system of claim 114 additionally comprising a plurality of document templates with components denoted thereon, whereby the browser code generated by the components can vary for different requests of the same document template (See Web Writer page 2 – 9, see editor, generator and browser and templates as well).

(11) Response to Arguments.

Claims 1 – 6, 8, 22 – 33, 41 – 43, 51 – 96, and 114 – 128 are rejected under 35 U.S.C. 103(a) as being unpatentable over Web Writer: A Browsers -Based Editor for Constructing Web Applications Arturo et al. Published May 1996 (Hereinafter “Web Writer”) in view of Web Writer II, Arturo et al. Published 1997.

Claim 1

Argument (b)

Appellant argues on page 19 of his Appeal brief, as Explained in sections A.ii and C.i, that Web Writer editor, “...doesn’t run the edited application during editing...”.

Response (b)

(1), Examiner still maintains that this limitation is taught by WebWriter.

Applicant's limitations as disclosed in claim 1, recites

".... document generator program running at least part of one of the applications being edited and generating the generated documents said generated documents including additional editing features for interpretation by the browser program (see Web Writer page Generator); and an editor program dynamically operating on the generated documents pages displayed by the browser program via the editing features (page 9, column 1, see Web Writer page Generator, see on the fly, also see page 2, first paragraph for buttons and computing content on the fly for components as claimed in claim 59).

In Webwriter under the heading Placing dynamic areas and WebWriter Page Generator:



Web writer discloses:

Creating the Template Pages

To create template pages, the designer uses the WebWriter Editor, which is a general-purpose HTML editor that runs in a Web browser. This editor has a two-column interface as shown in figure 2. The left column, called the *document area*, contains the page that is being edited. The right column contains a control panel that has three parts. The upper part provides file commands. The middle part provides

Placing Dynamic Areas

Dynamic areas are inserted into a template page to indicate regions of the page that will be filled in at runtime by a program. There are two types of dynamic area supported by WebWriter: *variable areas* and *output areas*. To create a variable area, the user simply types a name preceded by "[!" and followed by "]" in any text element of the document. For example, the user might add a string such as "Files in directory: [!directory]". The meaning of this string is that, at runtime, the string in square brackets will be replaced by the value of the variable "directory". Variables will be described in the next section.

An output area is a special WebWriter object (the only special tag that WebWriter adds to standard HTML). The user creates an output area by selecting Output from the Form menu (recall figure 4) and clicking Insert. WebWriter then adds a placeholder for the output area at the current insertion point. The output area object will be replaced at runtime by the output of a script, formatted as HTML and inserted at that point in the document. The creation form for each output area asks the user to specify a script to run. The contents of the output area (in between the  and  handles in the document area) specifies how the output of the script should be formatted into HTML. Both scripts and formatting are discussed in more detail below.

The WebWriter Page Generator

When a WebWriter application is running, each new page is assembled by the WebWriter Page Generator, another server-based CGI C++ program that knows how to read a template page, run the scripts specified in that template page and create a new page for display to the user. This Page Generator takes the place of the hand-written CGI program that would otherwise have to be produced to create a server-side Web application.

The Page Generator takes two arguments: the URL of a template page, and a list of variable=value pairs that represent the application variables as described in the section Constructing a WebWriter Application. The template page has been written in a version of HTML that has been extended in two ways: First, the template page may contain tags of the form:

```
<OUTPUT SCRIPT="...">formatting string</OUTPUT>
```

where each OUTPUT tag represents a WebWriter output area. At runtime, the entire output area is replaced by the HTML string that results from executing the program specified by the SCRIPT argument, and then converting it to HTML using the information in the formatting string. Second, the template page may contain strings of the form [!variable]. Such strings will not appear literally in the newly generated page. Instead they will be replaced by the value of variable, expressed textually, where the value of each "variable" is passed as part of the URL as described above.

The Page Generator generates a new page on the fly as it scans through the template page. As it reads, it copies each character to its output stream, until it encounters a [!variable] string or an <OUTPUT> tag.

"

As disclosed above and as recited in Applicant's claim. The document generator runs an **at least part** of the application being edited and that is interpreted by the browser.

WebWriter above shows the application is running while at least editing a part of the application during runtime. In the first Paragraph underneath the heading (Placing Dynamic Areas) WebWriter discloses, "Dynamic areas are inserted into a template page to indicate regions of the page that will be filled in at runtime (*i.e. at least part of the application being edited as claimed by Applicant*)..." (Emphasis added).

Under the heading creating the template pages, it is also taught that the created pages run in a Web Browser which also shows Applicants limitation of being interpreted by the Web Browser.

The edited program portion of Appellants claim, which discloses an editor program dynamically operating on the generated documents pages displayed by the browser, is also disclosed above under the heading, placing dynamic areas which clearly discloses areas of the code which are dynamically replaced at runtime called placeholders. Again Examiner maintains this limitations are taught by prior art.

Argument (c)

Appellant argues on page 20, that the prior art doesn't disclose any editing functions because it doesn't permanently modify the template but just temporarily transforms the page template into a generated page.

Response (c)

(1), Contrary, to Appellants assertion, Examiner maintains that the prior art does in fact performing editing functions and further more

Applicant's claim neither precludes or excludes his editing functionality as permanent or temporary and thus is arguing for an unclaimed merit of distinction which would render such an argument moot.

However, the limitation is still being taught by Prior art under the section, Web Editor, here's an exert from the article as recited below:

The next time WebWriter is called (after the user clicks on a handle or control panel button) the editor restores the state of all data structures from the information in the hidden tag. Next, the WebWriter performs the action requested by the user. There are three types of actions: changes to the document state, such as insert, select, modify, cut, copy, and paste; file operations such as save and load; and changes to the document view, such as hiding handles or showing handles.

We describe the implementation of these three action types by considering an example of each: Insert, Save, and HideHandles. When the user clicks on Insert, WebWriter inserts a new object of the requested class into the content tree data structure at the position designated by the insertion point and sets this new object to be the selected object. Then WebWriter generates a new page as described above: the new page will show the new object and its creation form. To Save, WebWriter asks each object to write an HTML version of itself to a file stream: WebWriter generates a new page including a message saying if the save succeeded or failed. To HideHandles, WebWriter changes the value of a Boolean variable to indicate that handles should be suppressed when each object generates its view in the new page: it then generates a new page without handles.

As disclosed the WebWriter is able to insert, modify, cut, copy and paste.

Argument (d)

Again Appellant argues that the editor program operating on the generated document via editing features isn't addressed.

Response (d)

Examiner disagrees. In an exert from Webwriter under the heading

WebWriter Editor. WebWriter discloses:

"...The WebWriter Editor is a server-based CGI service written in C++. Each time it is called, it generates a two-column page, like the one shown in figure 2. Each page contains a form and buttons which, when pressed, call the WebWriter Editor again to generate the next page.

Each page generated by the Editor contains HTML code that represents the appearance of the current document in the left column...

WebWriter performs the action requested by the user. There are three types of actions: changes to the document state, such as insert, select, modify, cut, copy, and paste; file operations such as save and load; and changes to the document view, such as hiding handles or showing handles...."



As noted Above the argued limitation by Appellant is indeed disclosed.

Further, regarding Appellants argument in the same section on page 23 first paragraph, Where Appellant rehashes previous arguments that

WebWriter runs the application after editing but does not run the application during editing. Examiner again points to the dynamic areas section previously recited;

Placing Dynamic Areas

Dynamic areas are inserted into a template page to indicate regions of the page that will be filled in at runtime by a program. There are two types of dynamic area supported by WebWriter: *variable areas* and *output areas*. To create a variable area, the user simply types a name preceded by "[!" and followed by "]" in any text element of the document. For example, the user might add a string such as "Files in directory: [!directory]". The meaning of this string is that, at runtime, the string in square brackets will be replaced by the value of the variable "directory". Variables will be described in the next section.

An output area is a special WebWriter object (the only special tag that WebWriter adds to standard HTML). The user creates an output area by selecting Output from the Form menu (recall figure 4) and clicking Insert. WebWriter then adds a placeholder for the output area at the current insertion point. The output area object will be replaced at runtime by the output of a script, formatted as HTML and inserted at that point in the document. The creation form for each output area asks the user to specify a script to run. The contents of the output area (in between the  and  handles in the document area) specifies how the output of the script should be formatted into HTML. Both scripts and formatting are discussed in more detail below.

It is disclosed above that the dynamic areas, are inserted with content at runtime by a program. As noted again, inserting is a form of editing and this is being performed at runtime as outlined in Webwriter.

Claims 6 and 8

Regarding claim 6, Examiner withdraws the previous rejection and claims are now allowable.

Claim 22

Claims 22 and previously addressed claim 1 differ only with regards to the inclusion of components.

Claim 22 discloses,

" An editor program operating with the web browser on generated documents and having instructions for inserting, deleting, and modifying components on document templates; and

a document generator program having instructions for processing document templates, for executing said components, and for generating the generated documents from the document templates that are understandable by the web browser..."

Claim 1 which was previously addressed shows the editor as well as document generator. The addition of "components" doesn't substantially change scope of the claims. As the recited components are still taught by WebWriter in the same recited sections.

Regarding inserting, deleting and modifying components, Examiner interprets those areas of the document to the equivalent to disclosure in WebWriter pertaining to The WebWriter Page Generator.

As noted Above WebWriter discloses a WebWriter page generator which also includes a templates page which includes variables and document portions in the template which Examiner interpreted to be Appellants components. As previously recited, Webwriter discloses:

"...WebWriter performs the action requested by the user. There are three types of actions: changes to the document state, such as insert, select, modify, cut, copy, and paste; file operations such as save and load; and changes to the document view, such as hiding handles or showing handles...."

There's also a section in WebWriter underneath the heading Defining Variables, which discloses user interface elements such as buttons, check boxes and radio buttons which Examiner still maintains to read on components.

Claim 26

In claim 26, Appellant argues that Webwriter doesn't show, " the second document appears and functions similar to the run-time view of the first document".

Examiner disagrees. Again Webriter discloses that:

"...The WebWriter Editor is a server-based CGI service written in C++. Each time it is called, it generates a two-column page, like the one shown in figure 2. Each page contains a form and buttons which, when pressed, call the WebWriter Editor again to generate the next page. Each page generated by the Editor contains HTML code that represents the appearance of the current document in the left column...(emphasis added)

As noted above, the generated page represents the appearance of the current documents, and as such Examiner maintains this is equivalent to Apellants above claimed limitation as argued in claim 26.

Claim 59

Regarding claim 59, Appellant is simply rehashing arguments already addressed above in claims 1, 22 and 26.

Claim 74 – 89, 90 – 96, and 114 - 128

Regarding claims Examiner withdraws the previous rejections and claims are not allowable.

Dependent Claim Arguments

Claim 2

Applicants argument regarding modifying components including features to insert, modify and delete have already been addressed above in argument response of claims 1, 22 and 26.

Claim 23

Regarding claim 23, the editing features argued by Appellant have already been addressed above in claims 1, 22 and 26 arguments, Examiner considers the editing mode to be part of the application editing which has already been addressed.

Claims 3 – 5, 24, 25, 27 – 29, 31, 43, 51 – 58, 64 – 66, and 70

Regarding claims Examiner withdraws the previous rejections and claims are now objected but would be allowable if the objected based claims

recited such subject matter as pointed out in the interview summary,
Pertaining to nested components or interactive components.

Claim 30

Adds the use of scripts and this is taught in WebWriter under the heading Placing Dynamic areas, scripts are recited.

Claim 33

Regarding the limitations in claim 33, in the section under the heading WebWriter editor, Webwriter discloses user requests which Examiner interprets to be Appellants argued limitation.

Claim 60

Regarding to Appellants argument that Webwriter's server side implementation cannot be combined with Webwriter II's client side portion Examiner, disagrees. As Webwriter II is an extension of Webwriter I and one would easily make that combination to take advantage of both implementations as described in Webwriter II's Abstract.

Claim 61, and 67 – 69

Examiner believes the editor portion is still being addressed in the response sections of 1, 22 and 26 above.

Claim 72

Regarding reloading the browser under the heading Webwriter Editor Webwriter saves and loads the templates, which Examiner interprets to equivalent to Appellants reloading the browser (i.e., template after its edited).

Claim 73

Regarding arguments in claim 73, claim still recites limitations pertaining to displaying and viewing edited content which was addressed above in responses to arguments in claims 1, 22 and 26 above.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Chuck Kendall.

September 13, 2010.

/Chuck O Kendall/
Primary Examiner, Art Unit 2192

Conferees

Tuan Dam, SPE AU 2192.

Gail Hayes, SPE 2100